Hierarchical Flow

Jeff Hartline[1]
jhartlin@cs.cornell.edu

Alexa Sharp[1]
asharp@cs.cornell.edu

[1] Department of Computer Science, Cornell University, Ithaca, NY 14853.

Abstract

This paper defines a hierarchical version of the maximum flow problem. In this model, the capacities increase over time and the resulting solution is a sequence of flows that build on each other incrementally. Thus far, hierarchical problems considered in the literature have been built on NP-complete problems. To the best of our knowledge, our results are the first to find a polynomial time problem whose hierarchical version is NP-complete. We present approximation algorithms and hardness results for many versions of this problem, and comment on the relation to multicommodity flow.

Keywords: hierarchical flow, hierarchical problems, incremental networks, maximum flow, multicommodity flow, concurrent flow, approximation algorithms

1 Introduction

There has been recent interest in hierarchical versions of classic problems such as facility location [1], clustering [2], and bin packing [3]. These problems model situations in which there is a natural hierarchy of levels with different characteristics, such as local vs. wide-area networks or multilevel memory caches. Hierarchical variations of NP-complete problems contain their non-hierarchical versions as special cases and therefore remain NP-complete. It is interesting to ask whether hierarchical versions of polytime problems remain polytime, or whether the hierarchical structure alters the problem enough to increase its complexity.

In this paper we study hierarchical versions of a classic polytime problem, the maximum flow problem. It turns out that the hierarchical variant is sufficiently different from the original problem to warrant independent study, and bears an interesting relationship to multicommodity flow. We present various complexity results concerning hierarchical flow.

A hierarchical flow problem is defined on a directed network $G = (V, E)$ with source s, sink t, and a non-decreasing sequence of k capacity functions $c_i : E \to \mathbb{Q}$, $1 \leq i \leq k$. The problem is to find k s-t flows f_i that optimize some objective function such that the flow f_i on any edge e does not exceed the capacity $c_i(e)$ but is at least $f_{i-1}(e)$, the amount sent along e by the previous flow.

Let $|f_i|$ denote the amount of f_i flow sent from s to t. For the *maximum sum flow* problem, the objective is to maximize the sum of the flows: $\max \sum_i |f_i|$. For the *maximum ratio flow* problem, given demand d_i for each level i, the objective is to satisfy the maximum possible proportion of each level's demand: maximize $r = \min_i \frac{|f_i|}{d_i}$.

One may notice similarities between the hierarchical flow and multicommodity flow (MCF) problems. The input to MCF is also a directed network $G = (V, E)$ but with a single capacity function $c : E \to \mathbb{Q}$ and k source-sink pairs $(s_i, t_i), 1 \leq i \leq k$. As in the hierarchical flow problem, there are two predominant MCF objective functions, and they correspond to the objectives stated above (*maximum multicommodity flow* and *maximum concurrent flow*, respectively).

Some of the results of this paper are in agreement with results for MCF [4, 5, 6]. For example, an optimal solution to the hierarchical flow problem is not necessarily integral, in contrast to the standard flow problem. Fractional solutions can usually be found in polynomial time, whereas finding an integral solution is NP-complete even in the simplest cases. We consider both directed and undirected graphs, simple (unit-capacity edges) and general graphs, and integral and fractional flows. We present hardness results and approximation algorithms for the maximum ratio problem. The results are summarized in Figure 1.

Any flow problem in which constraints rise over time and rerouting flow carries an implicit cost would benefit from this type of hierarchical model. The canonical example of flight scheduling, where discontinuing a flight leg is undesirable, is one of many applications for which the hierarchical model would be useful.

2 Hierarchical Flow

2.1 Hierarchical Flow Networks

A *hierarchical flow network* is a directed graph $G = (V, E)$ with a non-decreasing sequence of capacity functions $c_i : E \to \mathbb{Q}$, $1 \leq i \leq k$. In particular, $c_{i+1}(e) \geq c_i(e)$ for all edges e. We call each i a *level*.

A *hierarchical s-t flow* is a sequence of flows $f_i : E \to \mathbb{R}$ such that f_i is an s-t flow with respect to capacity function c_i (flow constraint) and $f_{i+1}(e) \geq f_i(e)$ for all levels i (hierarchical constraint). We denote the flow at level i by f_i, its value by $|f_i|$, and a maximum flow at level i by f_i^*. An $(a_1, a_2, \ldots, a_k)$ flow is a hierarchical flow having value a_i at level i, and a $(c_1, c_2, \ldots, c_k)$ edge is an edge having capacity c_i at level i. A $*$ in one of these entries denotes any possible value.

As an example, Figure 2(a) is a 2-level hierarchical flow network. We can achieve a $(0, 2)$ flow in this network by sending no level 1 flow and two units of level 2 flow in parallel along paths $s \cdot u \cdot t$ and $s \cdot v \cdot t$. Alternatively, we can achieve a $(1, 1)$ flow by sending one unit of level 1 flow along the only level 1 path $s \cdot u \cdot v \cdot t$ and no additional level 2 flow. No $(1, 2)$ flow exists because using uv in level 1 precludes us from sending any additional level 2 flow.

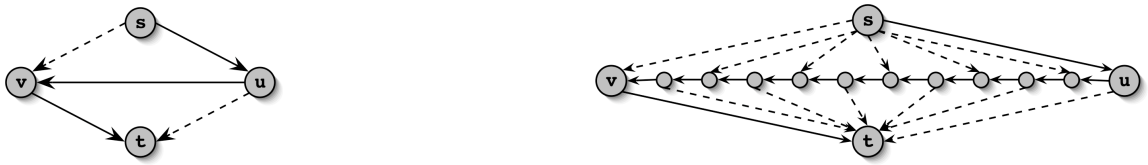
2.1.1 Undirected Networks

We also examine the hierarchical flow problem where the underlying network is undirected. In such cases we consider each undirected edge $e = \{u, v\}$ to be two directed edges $e = (u, v)$ and $\overline{e} = (v, u)$, both with capacity $c(u, v)$. Although the conservation constraints remain the same, there are two possible interpretations of the capacity constraints.

Bidirectional Constraints For all $e \in E$ and $1 \leq i \leq k$, $f_i(e) + f_i(\overline{e}) \leq c_i(e)$. One may send flow in both directions along an edge so long as the total amount being sent along the edge is less than the capacity of that edge.

	directed	bidirectional undirected	unidirectional undirected
Integral	NPC (S2) no approx. $> \mathrm{O}(\mathrm{n}^{-1})$ (S2*) $\mathrm{O}(\mathrm{n}^{-1})$ approx. alg. (G)	NPC (S2) no approx. $> \frac{1}{2}$ (S2*) no approx. $> \mathrm{O}(\mathrm{n}^{-1/3})$ (S3*)	
Fractional	LP solvable (G) $O(k^{-1})$ approx alg (G) PTAS (G)	LP solvable (G) $O(k^{-1})$ approx alg (G)	NPC (G3) no approx. $> \mathrm{O}(\frac{2}{3})$ (G3*) open (S2, G2, S3)

Figure 1: S: simple, G: general, 2: 2 levels, 3: 3 levels, *: unless $P = NP$



(a) (b)

Figure 2: Two example hierarchical flow networks. Throughout this paper we represent $(1, 1)$ edges with solid lines and $(0, 1)$ edges with dashed lines unless otherwise stated.

Unidirectional Constraints For all $e \in E$ and $1 \leq i \leq k$, $f_i(e) > 0 \Rightarrow f_j(\overline{e}) = 0$ for all $j \geq i$, and $\max\{f_i(e), f_i(\overline{e})\} \leq c_i(e)$. An undirected edge can carry flow in either direction but it must carry flow the same direction in all levels.

Observe that these constraints are equivalent in the integral unit-capacity case.

2.2 The Hierarchical Flow Problem

In contrast to the maxflow problem, where the single objective is to maximize flow into the sink, there are many possible objective functions in the hierarchical variation. Of the two mentioned in the introduction, this paper concentrates on the *max ratio problem.*

Given demands d_i, the max ratio problem asks to find a $(a_1, a_2, \ldots, a_k)$ flow maximizing $\min_i\{\frac{a_i}{d_i}\}$. We focus on the special case $d_i = |f_i^*|$, but all the results generalize. In this case, the problem is to satisfy the maximum possible proportion of each level's maximum flow: determine the largest ratio r^* such that $a_i \geq r^*|f_i^*|$ for all i. This ensures that adding a hierarchical aspect to the flow problem does not make any one level arbitrarily worse than it would have been without the hierarchical restrictions. This is a standard metric for hierarchical problems [1].

It is well known that for the maxflow problem with integer edge capacities, the optimal fractional flow is no better than the optimal integral flow. Interestingly enough, this does not carry over to the hierarchical version of the problem. Similar phenomena have been observed in the multicommodity flow problem.

Figure 2(a) shows it is not always possible to obtain a ratio of 1 at each level. In fact, the best we can do with integral flow is $r^* = \frac{1}{2}$. However, if we allow fractional flows we can obtain $r^* = \frac{2}{3}$ with a $(\frac{2}{3}, \frac{4}{3})$ flow. We generalize this example in Figure 2(b) to show that the best attainable guarantee with integral flows is $r^* = O(\frac{1}{n})$, where n is the number of vertices in the network. In this case $|f_1^*| = 1$ and $|f_2^*| = \frac{n}{2}$, but sending 1 unit of level 1 flow limits our level 2 flow to $|f_2| = 1 = \frac{2}{n}|f_2^*|$. With integral flow, this is the best ratio we can achieve. If we are willing to allow fractional flow, we can obtain $r^* = \frac{1}{2}(\frac{n}{n-1})$ on this network with a $(\frac{1}{2}(\frac{n}{n-1}), \frac{n}{4}(\frac{n}{n-1}))$ flow. These examples can be generalized further to show that for some k-level networks, $r^* = O(\frac{1}{k})$ even for fractional flow.

This paper investigates the integral and fractional cases separately. In Section 3 we discuss approximability and non-approximability results for integral hierarchical flow. Section 4 examines the fractional version analogously.

3 Integral Flows

We have observed that, in contrast to the maxflow problem, optimal hierarchical solutions are not always integral. Moreover, if we insist on integrality, then the hierarchical flow problem is NP-complete. Although this can be shown by reduction from MCF, we present

(a) Clause c (b) Variable v

Figure 3: Gadgets used in our directed flow construction. Literal gadgets appear inside labeled ovals. Observe that the same literal gadget (c, v) appears in both clause gadget c and variable gadget v.

a reduction from 3-SAT for the sake of obtaining stronger inapproximability results. We denote the decision version of the max ratio problem by r-ratio.

3.1 Hardness

3.1.1 Directed Flow

Theorem 3.1 *The r-ratio problem is NP-hard for directed graphs with integral flow.*

The proof of Theorem 3.1 follows from a reduction from 3-SAT. We construct an instance of the directed flow problem in which $|f_1^*| = 1$ and $|f_2^*| = 2$. For this instance, a $(1, 2)$ flow is a 1-ratio flow. We show that achieving such a flow is possible if and only if the 3-SAT instance has a satisfying assignment.

We are given an instance of 3-SAT with variables V and clauses C. Each clause is a set of three literals. The set of literals is the union of all positive literals v and all negative literals $\overline{v}$ for $v \in V$. We denote each literal occurrence as a clause-literal pair $(c \in C, \ell \in c)$.

Literal Gadgets For every (c, ℓ) we create a literal gadget: an *in* vertex, an *out* vertex, and a $(1, 1)$ directed link between these two vertices.

Clause Gadgets We create a clause gadget for every clause $c \in C$. Each clause gadget contains an *in* vertex, an *out* vertex, and the three literal gadgets of the form $(c, \ell \in c)$. These elements are linked together with $(1, 1)$ edges as shown in Figure 3(a).

Variable Gadgets We create a variable gadget for every variable $v \in V$. Each variable gadget consists of an *in* vertex, an *out* vertex, and all literal gadgets of the form (c, v) or $(c, \overline{v})$ for any c. These elements are linked together as shown in Figure 3(b): positive gadgets on one side and negative gadgets on the other, with each respective side linked in series with $(0, 1)$ edges.

Linkage We link the source, all clause gadgets, and the sink together in series with $(1, 1)$ edges. The same is done for the variable gadgets with $(0, 1)$ edges. We call these gadget-linking edges *connector* edges. Finally, we use a $(0, 1)$ *shortcut* edge to link the input node of the first variable gadget with the output node of the last clause gadget. Linkage is shown in Figure 4(a). Note that every literal gadget appears once in a clause gadget and once in a variable gadget, thus the seemingly separate source-sink paths in Figure 4(a) actually share many vertices.

Lemma 3.2 *The maximum single level flows have values $|f_1^*| = 1$ and $|f_2^*| = 2$.*

Proof. The sink has in-degree 1 in the level 1 graph and in-degree 2 in the level 2 graph, thus $|f_1^*| \leq 1$ and $|f_2^*| \leq 2$. A $(1, *)$ flow is achievable because the level 1 graph is connected. A $(*, 2)$ flow is achievable using the shortcut link as shown in Figure 4(b).

Lemma 3.3 *There is a satisfying assignment iff there exists a 1-ratio flow.*

[$\Rightarrow$] Given a satisfying assignment, we identify one true literal (c, ℓ) for each clause c. We route one unit of level 1 flow serially through all clause gadgets, passing through gadget c using literal (c, ℓ). We route one additional unit of level two flow serially through all variable gadgets, using only false literals by passing through v on the positive side if v is false and the negative side if v is true. Such a flow is shown in Figure 4(c).
[$\Leftarrow$] We first make the following observations concerning $(1, 2)$ flows in our construction:

1. The level 1 component of any $(1, 2)$ flow is a path including every $(1, 1)$ connector and one literal from each clause gadget. This is because the three literals of any clause form a level 1 cut, as does any $(1, 1)$ connector.

2. In any $(1, 2)$ flow, level 2 flow at $(c, v).in$ (or $(c, \overline{v}).in$) must proceed to $(c, v).out$ ($(c, \overline{v}).out$) and back to the positive (negative) side of variable gadget v. The only other path would be to $c.out$, whose sole out edge is a $(1, 1)$ connector already carrying level 1 flow (see observation 1).

3. In any $(1, 2)$ flow, level 2 flow at $v.in$ must proceed through either all (c, v) gadgets or all $(c, \overline{v})$ gadgets to $v.out$. This is because $v.in$ has two out edges: one to a sequence of all (c, v) gadgets and the other to a sequence of all $(c, \overline{v})$ gadgets. Both sequences terminate at $v.out$. Once flow proceeds to one of these sequences, induction on observation 2 implies it must pass through every literal gadget in the targeted sequence and end at $v.out$.

4. The level 2 component of any $(1, 2)$ flow is a path passing through the positive or negative side of each variable gadget. The source vertex has only two out edges: a $(1, 1)$ connector to a clause gadget and a $(0, 1)$ connector to a sequence of all variable gadgets. By observation 1, the $(1, 1)$ connector is already used, forcing the flow to the first variable gadget. Repeated applicaton of observation 3 implies that the flow proceeds through the positive or negative side of every variable gadget to the sink.

By observation 4, every variable v carries level 2 flow through one of its sides. We set v false if this flow passes through v's positive side and true otherwise. Under this assignment, all false literals carry level 2 flow. By observation 1, one literal from each clause must carry level 1 flow (and not level 2 flow) and thus cannot be false. This assignment satisfies 3-SAT.

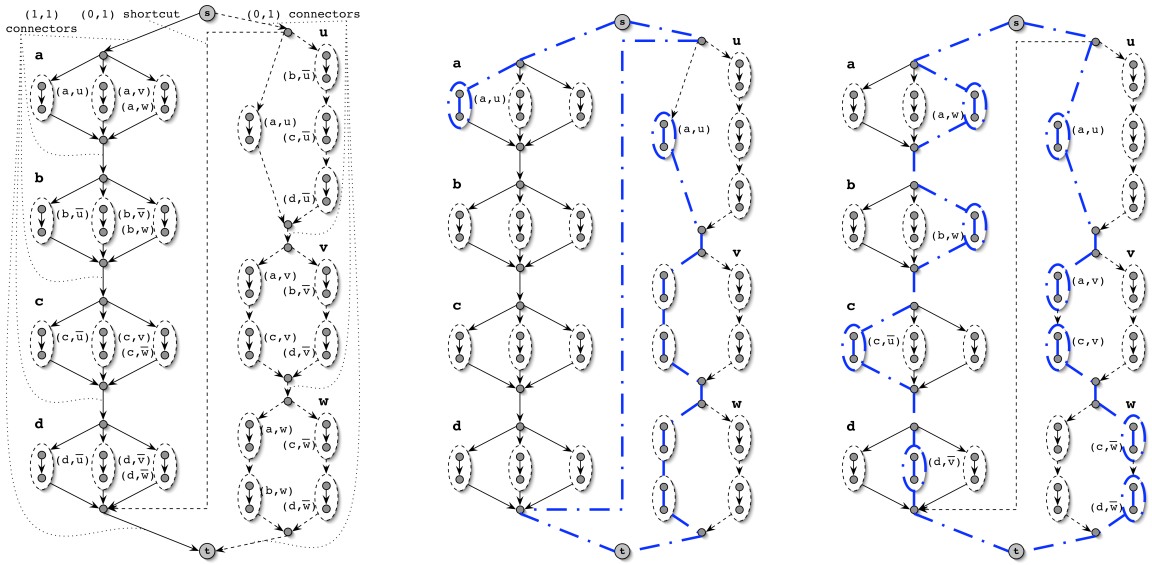
Theorem 3.1 follows from Lemmas 3.2-3.3 and the polynomial nature of our reduction.

3.1.2 Undirected Flow

Theorem 3.4 *The r-ratio problem is NP-hard for undirected graphs with integral flow.*

Proof. Theorem 3.4 follows from a slight modification of the reduction used in the directed case. If we simply take our directed construction and remove directionality then observations 1-4 of Lemma 3.3 [$\Leftarrow$] no longer hold. We recover equivalent observations by replacing each clause gadget with the undirected clause gadget shown in Figure 5(a) and doubling the number of $(1, 1)$ connectors. A $(2, 3)$ flow is a 1-ratio flow in the resulting construction.

We state without proof the following variation of observation 1, which can be used to establish observations 2-4 for $(2, 3)$ flows in the undirected construction. The level 1 component of any $(2, 3)$ flow specifies two paths that together use every $(1, 1)$ connector and at least one literal from each clause. Furthermore, this flow disconnects the unused literals from each other as illustrated in Figure 5(b). The rest of the proof follows as before.



(a) (b) (c)

Figure 4: Example construction and flows. Flow paths are shown as bold dashed lines. 4(a): Two-level flow reduction from 3-SAT for $EX = (u \vee v \vee w) \wedge (\overline{u} \vee \overline{v} \vee w) \wedge (\overline{u} \vee v \vee \overline{w}) \wedge (\overline{u} \vee \overline{v} \vee \overline{w})$. The clauses of EX are denoted a, b, c, d. Note that there are twelve distinct literal gadgets, each of which appears once in a clause gadget and once in a variable gadget. 4(b): A possible (*,2) flow using the shortcut edge. 4(c): A $(1, 2)$ flow based on the assignment $u = 0, v = 0, w = 1$.

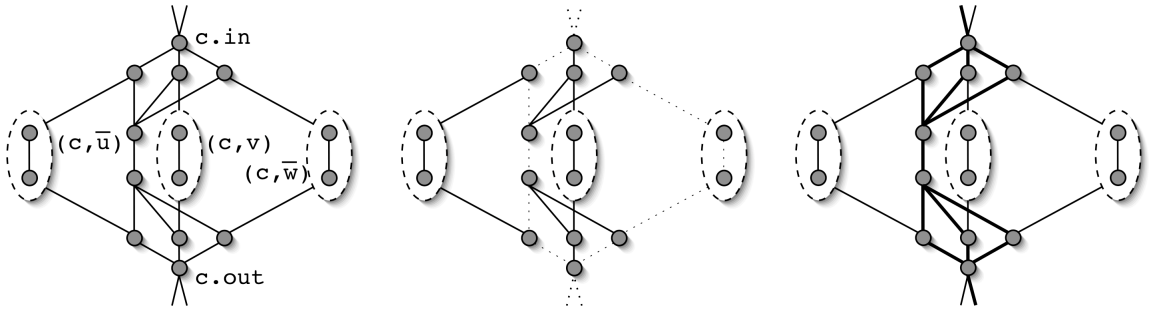
3.2 Inapproximability

3.2.1 Directed Flow

The following two theorems show we have a tight $O(\frac{1}{n})$-approximation algorithm for directed integral max ratio.

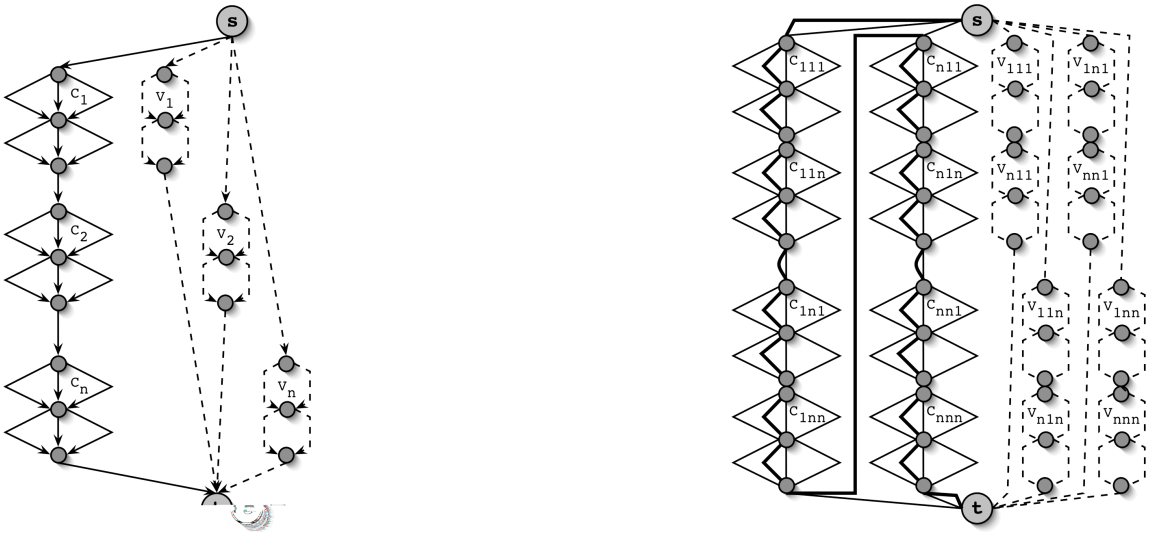
Theorem 3.5 *Directed integral max ratio is NP-hard to $g(n)$-approximate for $g \in \omega(\frac{1}{n})$.*

Proof. Given an instance of 3-SAT, we compose an instance of the directed flow problem out of N copies of the network constructed in Section 3.1.1. These copies are joined as shown in Figure 6(a) to create a network of size $n = \Theta(SN)$, where S denotes size of the original network. By the arguments of Lemma 3.3, there is a $(1, N + 1)$ flow iff there is a satisfying assignment. Furthermore, if there is no satisfying assignment then there is no (a_1, a_2) flow for $a_1 > 0$ and $a_2 > 1$. For any $g(n) \in \omega(\frac{1}{n})$, we can pick N sufficiently large to ensure that a $g(n)$-approximation distinguishes between these cases.



(a) (b) (c)

Figure 5: 5(a): An example undirected clause gadget for $c = \{\overline{u}, v, \overline{w}\}$. 5(b): One way to route two units of flow through this gadget. Any routing will disconnect the unused literal gadgets as demonstrated here. 5(c): The three level clause gadget used in Theorem 3.7. Bold lines represent level 0 links.



(a) Directed

(b) Undirected

Figure 6: Constructions for non-approximation results for integral max ratio. Both of these constructions paste together the clause and variable components of many copies of the constructions described in Section 3.1. We label clause and variable components of copy x as C_x and V_x, respectively. Level $0, 1, 2$ edges are shown as bold, solid, and dashed lines.

Theorem 3.6 *There is a greedy $O(\frac{1}{n})$-approximation for directed integral max ratio.*

Proof. Consider the algorithm that starts at level 1 and sequentially sends the maximum flow possible on each level, given the hierarchical constraint imposed by the previous flow. For any level i, let P be a single unit flow path of the greedily constructed f_i. Each edge of P could have been used by at most one unit flow path of each f_j^*, $j \geq i$, and since $length(P) < n$, the path P could have blocked at most $n-1$ units of f_j^*'s flow. Therefore

$$\begin{aligned} |f_j| &= |f_j^*| - \text{number of paths blocked by } f_{j-1} \\ &\geq |f_j^*| - (n-1)|f_{j-1}| \quad \geq \quad |f_j^*| - (n-1)|f_j| \quad \geq \quad \frac{1}{n}|f_j^*|. \end{aligned}$$

3.2.2 Undirected Flow

Theorem 3.7 *Two level undirected integral max ratio is NP-hard to α-approximate for $\alpha > \frac{1}{2}$. For three or more levels it is NP-hard to $g(n)$-approximate for $g \in \omega(n^{-1/3})$.*

Proof. The two-level result follows from a modification of the reduction used in the directed case (Theorem 3.5) where we remove directionality as described in Section 3.1.2. We obtain the stronger 3-level approximation result by adding a level to our clause gadget as shown in Figure 5(c) and enhancing the construction as follows:

Given an instance of 3-SAT, we compose an instance of the undirected flow problem out of N^3 copies of the network constructed in Section 3.1.2 using multi-level clause gadgets. These copies are joined as shown in Figure 6(b) to create a network of size $n = \Theta(SN^3)$, where S denotes size of the original network. There is a $(1, N+1, N^2+N+1)$ flow iff there is a satisfying assignment. Furthermore, if there is no satisfying assignment then there is no (a_0, a_1, a_2) flow for $a_0 > 0$, $a_1 > 1$, and $a_2 > N+1$. For any $g(n) \in \omega(n^{-1/3})$, we can pick N sufficiently large to ensure that a $g(n)$-approximation distinguishes between these cases.

4 Fractional Flows

Theorem 4.1 *Directed fractional max ratio and bidirectional fractional max ratio are poly-time.*

Proof. We can formulate the directed fractional max ratio problem as a linear program (LP) in $O(mk)$ variables and constraints. The objective is to maximize r such that $f_i(e)$ and r are non-negative, $f_{i-1}(e) \leq f_i(e) \leq c_i(e)$, and

$$\begin{aligned} \sum_{e \text{ out of } v} f_i(e) - \sum_{e \text{ into } v} f_i(e) &= 0 \\ \sum_{e \text{ out of } s} f_i(e) - \sum_{e \text{ into } s} f_i(e) &\geq r \cdot d_i. \end{aligned}$$

For the bidirectional case, we have the additional constraint that $f_i(e) + f_i(\overline{e}) \leq c_i(e)$.

Unfortunately, it is not known how to formulate an LP for the unidirectional case, as we have the non-linear constraint $f_j(e) > 0 \rightarrow f_i(\overline{e}) = 0$ for all $i \geq j$. In fact, the following result shows that it is not possible to do so unless P=NP.

Theorem 4.2 *The r-ratio problem is NP-hard in the unidirectional fractional case.*

Proof. We begin with the complete construction of Section 3.1.1 as shown in Figure 4(a). We call all edges not within a literal gadget *auxiliary* edges. Denote the first and last variables in the construction v_{first} and v_{last}, respectively, and the last clause as c_{last}. Modify the construction as follows. Unidirectional clause and variable gadgets are shown in Figure 7.

1. Replace $(s, v_{first}.in)$, $(v_{last}.out, t)$, and shortcut $(v_{first}.in, c_{last}.out)$ with $(t, v_{first}.in)$, $(v_{last}.out, s)$, and $(c_{last}.out, v_{first}.in)$. Give these new edges capacity $(0, 1)$.

2. Subdivide each auxiliary edge into three sequential edges. Color the middle edge *blue*.

3. Create a single s-t path that passes forward through all $(1, 1)$ blue edges and backwards through all $(0, 1)$ blue edges by linking them in series with blue $(0, 0)$ links. These edges will get positive capacity later.

4. Add a 0^{th} level: replace all blue (c_1, c_2) edges with $(\epsilon, c_1+\epsilon, c_2+\epsilon)$ links and all normal (c_1, c_2) edges with $(0, c_1, c_2)$ links.

5. Remove link directionality to yield an undirected graph.

This modified construction has optimal single-level flows $|f_0^*| = \epsilon$, $|f_1^*| = 1 + \epsilon$, and $|f_2^*| = 2 + \epsilon$. Arguments similar to those used in Lemma 3.3 show that a satisfying assignment can be used to achieve a $(\epsilon, 1 + \epsilon, 2 + \epsilon)$ flow. Similarly, such a flow f can be used to determine a satisfying assignment. This is because $|f_0| = \epsilon$ implies that all blue edges carry level 0 flow in the direction specified by (3). Furthermore, $|f_1| = 1 + \epsilon$ implies a total of one unit of flow passes down through the three literals in every clause, and thus at least one literal per clause is directed down. Lastly, $|f_2| = 2 + \epsilon$ implies that a unit of flow goes up through every variable gadget. For reasons similar to those discussed in Theorem 3.1 this flow must proceed through all positive literals or negative literals (or both, if we split the flow) for each variable v. Note that no level 2 flow can proceed up through (v, ℓ) if any amount of level 1 flow went down through (v, ℓ). We assign v false if any of this level 2 flow goes up through v's positive literals and true otherwise. For reasons analogous to those discussed in Section 3.1.1 such an assignment necessarily solves the 3-SAT instance.

Figure 7: Three-level clause and variable gadgets for the unidirectional construction. Thin blue lines are $(\epsilon, \epsilon, \epsilon)$ links. Solid blue lines are $(\epsilon, 1+\epsilon, 1+\epsilon)$ links. Dashed blue lines are $(\epsilon, \epsilon, 1+\epsilon)$ links. Solid and dashed black lines are $(0, 1, 1)$ and $(0, 0, 1)$ links, respectively.

4.1 Approximation Algorithms

We can use LP techniques to find exact solutions for the directed and bidirectional max ratio problem. Unfortunately, current LP techniques are impractical, leading us to search for faster combinatorial algorithms to solve the problem exactly, and that failing, faster combinatorial algorithms to solve the problem approximately.

A polytime algorithm for the k-level fractional directed max ratio problem also solves the k-commodity concurrent flow problem. There are no currently known combinatorial algorithms for the k-commodity concurrent flow problem for $k \geq 3$. Therefore we focused our efforts on the 2-level problem. In this we were unsuccessful, despite attempts at generalizing 2-commodity concurrent flow techniques [7, 4]. Although we did not find an exact solution, we did find a straightforward $\frac{1}{k}$-approximation for the directed and bidirectional cases. We start with a level 1 flow of f_1^*/k, and for each subsequent level i we define $f_i = f_{i-1} + f_i^*/k$.

In the directed case we can do better. There is a polytime approximation scheme (PTAS) for max ratio based on the techniques of [8, 9, 10]. This algorithm can be motivated by the dual of the path-flow linear programming formulation of the max ratio problem: for each level ℓ, the algorithm assigns a length $y_\ell(e)$ to each edge e. For an s-t path P, let $w_\ell(P)$ denote the cost of P, where the cost of edge e is $w_\ell(e) = \sum_{j \geq \ell} y_j(e)$.

Initially the algorithm sets $y_\ell(e) = \delta/k c_\ell(e)$ and $f_\ell \equiv 0$ for all levels ℓ. It then proceeds in phases; each phase is composed of k iterations. In the j^{th} iteration, the objective is to route $d_j - d_{j-1}$ units of level j flow from s to t. This is done in steps. In each step, a least-cost path P in the level j graph is computed using the cost function w. Let b be the minimum of the bottleneck capacity of P and the remaining demand. We send b units of flow along P and for each edge $e \in P$ and level $\ell \geq j$ we set $f_\ell(e) = f_\ell(e) + b$ and update the length function $y_\ell(e) = y_\ell(e)(1 + \epsilon \frac{b}{c_\ell(e)})$. The entire procedure stops when $\sum_{\ell=1}^{k} \sum_e c_\ell(e) y_\ell(e) \geq 1$.

Theorem 4.3 *A $(1-\epsilon)^3$-approximate solution to the k-level hierarchical max ratio flow problem can be obtained in $\tilde{O}(\epsilon^{-2} \log n(m+n)(k+m))$ time.*

The proof follows with modifications from [8, 10].

References

[1] C. G. Plaxton, "Approximation algorithms for hierarchical location problems," in *Proceedings of the thirty-fifth annual ACM symposium on Theory of computing*. ACM Press, 2003, pp. 40–49.

[2] S. Dasgupta, "Performance guarantees for hierarchical clustering," in *Proceedings of the 15th Annual Conference on Computational Learning Theory*. Springer-Verlag, 2002, pp. 351–363.

[3] B. Codenotti, G. D. Marco, M. Leoncini, M. Montangero, and M. Santini, "Approximation algorithms for a hierarchically structured bin packing problem," *Inf. Process. Lett.*, vol. 89, no. 5, pp. 215–221, 2004.

[4] T.C.Hu, "Multi-commodity network flows," *Operations Research*, vol. 11, no. 3, pp. 344–360, 1963.

[5] S. Even, A. Itai, and A. Shamir, "On the complexity of timetable and multicommodity flow problems," *SIAM Journal on Computing*, vol. 5, no. 4, pp. 691–703, 1976.

[6] T. Leighton, C. Stein, F. Makedon, É. Tardos, S. Plotkin, and S. Tragoudas, "Fast approximation algorithms for multicommodity flow problems," in *Proceedings of the twenty-third annual ACM symposium on Theory of computing*. ACM Press, 1991, pp. 101–111.

[7] A. Itai, "Two-commodity flow," *J. ACM*, vol. 25, no. 4, pp. 596–611, 1978.

[8] N. Garg and J. Koenemann, "Faster and simpler algorithms for multicommodity flow and other fractional packing problems." in *Proceedings of the 39th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1998, p. 300.

[9] L. K. Fleischer, "Approximating fractional multicommodity flow independent of the number of commodities," *SIAM J. Discret. Math.*, vol. 13, no. 4, pp. 505–520, 2000.

[10] K. D. Wayne and L. Fleischer, "Faster approximation algorithms for generalized flow," in *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1999, pp. 981–982.